UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/862,828	05/22/2001	Neil W. Taylor	971-128	8874

7590 11/14/2008

MICHAEL T. SANDERSON, ESQ
KING & SCHICKLI, PLLC
247 NORTH BROADWAY
LEXINGTON, KY 40507

EXAMINER	
SONG, HOSUK	

ART UNIT	PAPER NUMBER
2435	

MAIL DATE	DELIVERY MODE
11/14/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

UNITED STATES PATENT AND TRADEMARK OFFICE

____

BEFORE THE BOARD OF PATENT APPEALS
AND INTERFERENCES

____

Ex parte NEIL W. TAYLOR

____

Appeal 2007-3498
Application 09/862,828
Technology Center 2100

____

Decided: November 14, 2008

____

Before JAMESON LEE, RICHARD TORCZON and SALLY C. MEDLEY, *Administrative Patent Judges*.

MEDLEY, *Administrative Patent Judge*.

DECISION ON APPEAL

A. Statement of the Case

Novell, Inc. ("Novell"), the real party in interest, seeks review under 35 U.S.C. § 134(a) of a Final Rejection of claims 1-4 and 7-22, the only claims remaining in the application on appeal. We have jurisdiction under 35 U.S.C. § 6(b). We affirm-in-part.

Novell's invention is related to a method of detecting when executable code has been altered. When the executable code is initially loaded into an operating system, an initial score is calculated. Subsequent scores on the executable code are calculated at a later time. If a subsequent score does not match the initial score, the executable code has been altered. If the executable code has been altered, it can be disabled and a user can be notified. Abs., Spec. 4-6, 14-15.

Representative claim 1, reproduced from the Claim Appendix of the Appeal Brief, reads as follows:

> A method for validating executable code resident in an operating system having executable instructions, comprising the steps of:
>
> identifying an executable code having an unaltered size;
>
> calculating an initial score associated with the executable code when the executable code is initially or shortly thereafter loaded into an operating system;
>
> saving the initial score;
>
> one of randomly and substantially each time the executable code is launched for use, calculating a plurality of subsequent scores on the executable code;
>
> exclusively comparing each of the subsequent scores to the saved initial score and to no other scores;
>
> if the each of the subsequent scores do not vary from the saved initial score, concluding the executable code maintains the unaltered size; and
>
> if any of the subsequent scores vary from the saved score, concluding the executable code has an altered size.

The Examiner relies on the following prior art in rejecting the claims on appeal:

Slivka et al. ("Slivka")	5,493,649	Feb. 20, 1996
Angelo	5,944,821	Aug. 31, 1999

The Examiner rejected claims 1-4 and 7-22 under 35 U.S.C. § 103(a) as unpatentable over Slivka and Angelo.

B. Findings of Fact ("FF")

<u>Slivka</u>

1. Slivka describes a method of detecting corruption of computer programs due to "bugs" in computer programs. Abs.; col. 1, l. 13-col. 2, l. 41.
2. Referring to figures 4A and 4B [numbers from figures 4A, 4B inserted], upon initially starting a file management component (e.g., system startup), the file management component of the operating system calculates a checksum for the code section of the file management component [402]. Col. 2, ll. 47-54, 60-65; col. 3, ll. 30-35.
3. The file management component waits until it receives an operation request to perform [406]. Col. 3, ll. 35-37.
4. Following the operation request to perform [406] there is a check to see if a periodic interval has elapsed [408]. Col. 3, ll. 37-45.
5. Modifications to the code section are very rare, and are preferably not checked upon receipt of every operation request. Col. 3, ll. 40-45.
6. The periodic interval can be based on timing, count of operations requested, or other suitable events. Col. 3, ll. 45-46.
7. If the periodic interval has not elapsed, the file management component skips steps [410] and [412] and continues to step [416] Col. 3, ll. 46-48.

8. If the periodic interval has elapsed, the file management component calculates a new code checksum [410]. Col. 3, ll. 48-51
9. The file management component determines whether both checksums are equivalent [412], and if they are equivalent processing continues to step [416]. Col. 3, ll. 58-59
10. If the checksums are not equivalent, the file management component indicates to the user that the code section has been corrupted and ends the processing before the code is executed [428]. Col. 3, ll. 54-58.
11. Access to the data is granted [422] if the new data checksum [416] is equivalent [418] to an initial data checksum [404]. Col. 3, ll. 30-35, 61-66; col. 4, ll. 3-14.
12. Following access to the data [422], the process continues back to receive a requested operation [406]. Col. 4, ll. 14-15.

Slivka's figures 4A and 4B are reproduced below.

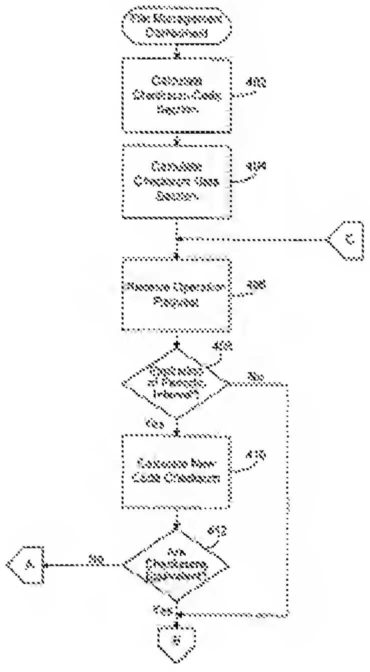

FIG. 4A

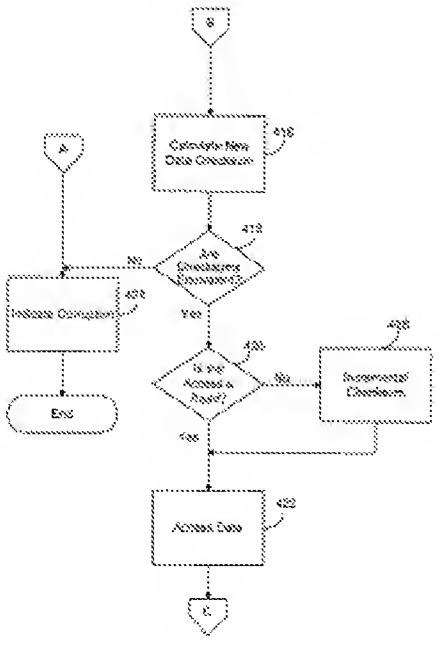

FIG. 4B

Figures 4A and 4B depict a flow chart for detecting corrupted code.

<u>Angelo</u>

13. Angelo is directed to assessing computer software code integrity. Abs.
14. Integrity and trustworthiness have little to do with bugs in the software, although bugs could jeopardize the integrity of software. Col. 1, ll. 49-64.
15. The problem of malicious code or computer viruses is a much greater threat to software code. Col. 2, ll. 8-10.
16. A virus is activated when the file is executed and may lay dormant. Col. 2, ll. 24-26
17. Angelo describes that its invention incorporates the capability to protect against the execution of unauthorized or modified code in real time. Col. 4, ll. 24-29.
18. A secure hash value is generated for a piece of software when it is installed on the computer system. Col. 4, ll. 45-50; col. 10, ll. 18-20.
19. Following hash value generation for newly-installed software, the hash values are stored in a secure hash table. Col. 4, ll. 48-50.
20. A System Management Interrupt (SMI) is generated when a user attempts to execute the program. Col. 4, ll. 55-57; col. 9, ll. 49-51; fig. 3, step 300.
21. The SMI handler generates a current hash value for the program to be executed which is compared with the stored hash value entry in the table. Col. 4, ll. 59-64; col. 9, l. 66-col. 10, l. 2; fig. 3, step 304.
22. If the hash values match, the program is loaded into memory and executed. Col. 3, ll. 64-67; col. 10, ll. 22-26; fig. 3, steps 312, 318.

C. Principles of Law

"In the patentability context, claims are to be given their broadest reasonable interpretations." *In re Van Geuns,* 988 F.2d 1181, 1184 (Fed. Cir. 1993) (citation omitted).

> A reference may be said to teach away when a person of ordinary skill, upon reading the reference, would be discouraged from following the path set out in the reference, or would be led in a direction divergent from the path that was taken by the applicant. The degree of teaching away will of course depend on the particular facts; in general, a reference will teach away if it suggests that the line of development flowing from the reference's disclosure is unlikely to be productive of the result sought by the applicant.

In re Gurley, 27 F.3d 551, 553 (Fed. Cir. 1994).

"Non-obviousness cannot be established by attacking references individually where the rejection is based upon the teachings of a combination of references." *In re Merck & Co., Inc.*, 800 F.2d 1091, 1097 (Fed. Cir. 1986).

Argument of counsel cannot take the place of evidence lacking in the record. *Meitzner v. Mindick*, 549 F.2d 775, 782 (CCPA 1977); *see also In re Pearson*, 494 F.2d 1399, 1405 (CCPA 1974).

"[T]he Federal Circuit has employed . . . the 'teaching, suggestion, or motivation' (TSM) test, under which a [ ] claim is only proved obvious if 'some motivation or suggestion to combine the prior art teachings' can be found in the prior art, the nature of the problem or the knowledge of a person having ordinary skill in the art." *KSR Int'l Co. v. Teleflex Inc.,* 127 S.Ct. 1727, 1734 (2007).

In *KSR*, the Supreme Court rejected the rigid application of the "teaching suggestion, or motivation" (TSM) test, instead favoring the "expansive and flexible approach" used by the Court. *Id.* at 1739.

"In determining whether the subject matter of a [ ] claim is obvious, neither the particular motivation nor the avowed purpose of the [applicant] controls. What matters is the objective reach of the claim. If the claim extends to what is obvious, it is invalid under § 103." *Id.* at 1741-2. "The question is not whether the combination was obvious to the [applicant] but whether the combination was obvious to a person with ordinary skill in the art. Under the correct analysis, any need or problem known in the field of endeavor at the time of the invention and addressed by the patent can provide a reason for combining the elements in the manner claimed." *Id.* at 1742.

D. Analysis

Claims 1-4, 7-10, 13-18 and 20-22

Claims 1-4, 7-10, 13-18 and 20-22 stand or fall together. Br. 2, 29-35. We focus our analysis on independent claim 1 since Novell's arguments are focused on the limitations of claim 1. Br. 18-21. Claim 1 recites "calculating an initial score associated with the executable code . . . one of randomly and substantially each time the executable code is launched for use, calculating a plurality of subsequent scores on the executable code; exclusively comparing each of the subsequent scores to the saved initial score and no other . . .". Br. 29. We interpret the claim limitations "one of randomly and substantially each time the executable code is launched for use, calculating a plurality of subsequent scores on the executable code . . ."

to mean that the plurality of subsequent scores are calculated either randomly or substantially every time the code is launched.

First, Novell argues that Slivka does not describe calculating an initial score and a plurality of subsequent scores and exclusively comparing each of the subsequent scores to the initial score and to no other scores. Br. 18-19. Novell argues that Slivka instead only describes a one-to-one comparison of a first code checksum to a new code checksum. Br. 18-19. Novell argues that Slivka does not have multiple instances of subsequent checksums or exclusive comparisons between the multiple instances of subsequent checksums to the first checksum and no others. Br. 18-19.

We are unpersuaded by Novell's arguments because non-obviousness cannot be established by attacking the references individually. Novell's arguments are directed to the Slivka reference alone. The rejection is based on the combination of references. Novell does not address the combination of Slivka and Angelo.

As the Examiner explains, Slivka describes performing the process of figures 4A and 4B upon initially starting the file management component code. Final Rejection 2-3; Ans. 3-4, 7-9. The Examiner further explains that Angelo describes calculating a score or checksum substantially each time the executable code is launched for use. Final Rejection 3; Ans. 4, 9, citing col. 9, l. 45-col. 10, l. 35. The Examiner determined that it would have been obvious to one with ordinary skill in the art at the time the invention was made to modify Slivka to incorporate Angelo's secure execution of the executable code every time it is launched in order to provide a safestart every time the program is executed. Thus, Slivka as modified by Angelo results in performing the process of figures 4A and 4B each time the file management component code is launched.

The result of the Slivka and Angelo combination is that each time the file management code (i.e., executable code) is launched there is a calculation [402] of a first code checksum (i.e., initial score). FF[1] 2. At some later point in time a new code checksum (i.e., subsequent score) is calculated [410]. FF 8. The timing of the calculation of the new code checksum is dependent upon the receipt of an operation request [406] and the expiration of the periodic interval [408]. FFs 3-5, 8. Even if the first instance of the periodic interval has not elapsed (no branch of [408]), the return of the process to step [406] in the continuous loop depicted in figures 4A and 4B will eventually result in a calculation [410] of a new code checksum (i.e., subsequent score). FFs 7, 9-12. Therefore, when the file management code (i.e., executable code) is launched a plurality of times, there is a plurality of resulting first code checksums (i.e., initial scores) and a plurality of resulting new code checksums (i.e., subsequent scores) and a plurality of comparisons between the respective first code checksums (i.e., initial scores) and the new code checksums (i.e., subsequent scores).

Second, Novell argues that even if Slivka calculates a third or more code section checksum, Slivka describes a periodic interval routine for introducing a delay into the code section comparison routine that prevents code section checksum comparisons from occurring *all the time*. Br. 17, 20-21. Novell further argues that Slivka's delay is regular or periodic during *all* code checksum comparisons. Br. 21.

[1] FF denotes Finding of Fact.

We agree that Slivka describes a periodic interval at step [408]. FFs 4-6. However, we are not persuaded by Novell's arguments because they are not commensurate in scope with the claim limitations. Claim 1 does not require code section comparisons to occur all the time. Br. 29. Claim 1 also does not require a time limit for the calculation of the plurality of subsequent scores or their comparisons. Br. 29. Instead, claim 1 only requires "calculating a plurality of subsequent scores" "one of randomly and substantially each time the executable code is launched for use" and also "exclusively comparing each of the subsequent scores to the saved initial score and to no other scores". Br. 29.

Related to the preceding argument, Novell argues that all of the claims require calculating plural subsequent scores nearly each time the executable code is launched for use or alternatively calculating the plurality of subsequent scores randomly. Br. 21. Novell argues that Slivka's checking for code modifications is only worthy of receiving periodic interval checksum comparisons. Br. 21.

Again, we are unpersuaded by Novell's arguments because they are directed to the Slivka reference alone rather than the combination of Slivka and Angelo. Non-obviousness cannot be established by attacking the references individually. Angelo was relied upon for describing calculating a score or checksum substantially each time the executable code is launched for use. Final Rejection 3; Ans. 4, 9, citing col. 9, l. 45-col. 10, l. 35. As explained before, the combination of Slivka and Angelo results in performing Slivka's process of figures 4A and 4B each time the file management component code is launched.

Further, Slivka describes introducing the periodic interval so that the code checksums are not calculated and compared each time there is a *request for operation* [406]. FF 5. The Examiner finds that the launching executable code reads on Slivka's description of *starting the file management component and its code*. Ans. 7-8. Therefore, Slivka's request for operation [406] is different from the starting of the file management component and its code. Slivka's operation requests [406] occur when the file management component code has already been launched.

Third, Novell argues that Angelo does not describe scores or calculations of scores associated with the executable code at a time when the code is initially or shortly thereafter loaded into an operating system. Br. 22. Novell argues that Angelo instead describes hash value generation and comparison <u>at a time prior to or before</u> the executable code is loaded or installed into an operating system. Br. 22-23 (emphasis in original).

Again, we are unpersuaded by Novell's arguments because they are directed to the Angelo reference alone rather than the combination of Slivka and Angelo. Non-obviousness cannot be established by attacking the references individually. Moreover, the Examiner does not rely on Angelo for describing these claim limitations, but instead relies on Slivka. Final Rejection 2-3, Ans. 4; citing Slivka col. 3, ll. 15-43.

Fourth, Novell argues that Angelo teaches away from both Slivka's invention and Novell's invention because it disparages solutions that attempt to solve problems of code modification detection via software products and/or code modification detection stored in other than protected memory areas. Br. 23-24, citing Angelo col. 2, l. 50-col. 3, l. 8. Novell also argues that Angelo teaches away because software products desire operation as part of regular routine modes of operation independently of SAFESTART. Br. 24.

We are not persuaded by Novell's arguments because Novell has not explained why a person of ordinary skill in the art upon reading Angelo would be discouraged from following the path set out in Angelo or be led in a direction divergent from the path that was taken by Novell.

Fifth, Novell argues that the Examiner's rejection is based on hindsight reconstruction of the claims and is a piecemeal and selective culling of bits and pieces of the references without any motivation to do so. Br. 25. Novell also draws attention to the fact that the instant application was filed over five years ago on May 22, 2001. Br. 25.

We are unpersuaded by Novell's arguments because the Examiner identifies that the motivation to combine the references is to provide a safestart every time a program is executed. Final Rejection 3, Ans. 5. Moreover, Novell has not explained the relevance of its filing date. In other words, Novell has not shown with supporting evidence, that one of ordinary skill in the art would not have known to combine the teachings of Slivka and Angelo prior to its filing date. The determination of obviousness is based on whether the invention would have been obvious to one with ordinary skill in the art at the time the invention was made. Novell has not directed us to evidence to support their argument that the Examiner used hindsight. Argument of counsel can not take the place of evidence lacking in the record.

Sixth, Novell argues that the Examiner has not met his burden of establishing obviousness because the Examiner's motivation to combine Slivka and Angelo does not relate to the nature of the problem to be solved.

Br. 26. Novell argues that Slivka already accounts for its user-based scenarios of computer mode operation and need not look to Angelo for any reason, much less a teaching of a safestart. Br. 26. Novell also argues that their invention does not address solving a problem of making sure safestart modes of operation have no integrity issues. It appears that Novell is arguing that there is no "teaching, suggestion, or motivation."

The rigid application of the "teaching, suggestion or motivation" test has been rejected by the Supreme Court. The Examiner identifies providing a safestart every time a program is executed as a need in the field of endeavor which provides a reason for combining the elements. Final Rejection 3, Ans. 4-5. Slivka addresses corruption of computer programs due to "bugs" in computer programs. FF 1. Slivka does not appear to contemplate corruption of computer programs due to malicious code or viruses. Angelo describes that the problem of malicious code or viruses is a much greater threat to software code integrity than computer bugs. FFs 13-15. Angelo describes that a virus is activated when the file is executed. FF 16. Angelo describes that its invention incorporates the capability to protect against the execution of unauthorized or modified code in real time. FF 17. Angelo describes checking the software integrity each time before execution of the software code. FFs 17-21.

We find that upon reading the Angelo reference, one with ordinary skill in the art would appreciate that malicious code and viruses pose a greater threat to program file corruption than computer program bugs. We further find that one with ordinary skill in the art would appreciate that viruses are not activated until execution of the program file. Still further, we find that one with ordinary skill in the art would appreciate that checking the program file integrity prior to execution every time the program file is to be executed is necessary to protect the program against threats posed by malicious code and viruses.

Therefore, we agree with the Examiner that it would have been obvious to one with ordinary skill in the art at the time the invention was made to modify Slivka's invention to incorporate Angelo's secure execution of the executable code every time it is launched in order to provide a safestart every time the program is executed. Providing a safestart every time a program is executed is beneficial to protect against malicious code and viruses which are not activated until the program file is executed.

For all these reasons, we find that Novell has not sustained its burden of showing that the Examiner erred in rejecting claims 1-4, 7-10, 13-18 and 20-22 as obvious over Slivka and Angelo.

<u>Claims 11 and 12</u>

Claims 11 and 12 stand or fall together. Br. 21-22. Claim 11 depends on claim 8 and further includes that "the subsequent scores are calculated randomly and substantially each time the executable code is launched for use." Br. 21.

In addressing claims 11 and 12, the Examiner finds that Slivka discloses receiving one or more additional scores periodically on the executable code and disabling the executable code if any of the subsequent scores are not equal. Final Rejection 5, Ans. 6; citing Slivka col. 3, ll. 40-60. The Examiner's findings are not commensurate in scope with the claim limitations. Since Slivka does not describe calculating the subsequent scores both randomly and substantially each time the executed code is launched and since we find no reason why a person of ordinary skill in the art would want to calculate scores both randomly and substantially each time the code is launched, we reverse as to claims 11 and 12.

Claim 19

Claim 19 stands or falls alone. Br. 21-22. Claim 19 recites "the subsequent scores are received each time the executable code is initiated in the memory for an execution." Br. 34. Novell argues that Slivka does not describe that subsequent scores need to be received "every time" the executable code is launched for use.

Again, we are not persuaded by Novell's arguments because they are directed to the Slivka reference alone rather than the combination of Slivka and Angelo. Non-obviousness cannot be established by attacking the references individually. As explained before, the combination of Slivka and Angelo describes that the subsequent scores are received each time the executable code is launched for use.

For all these reasons, we find that Novell has not sustained its burden of showing that the Examiner erred in rejecting claim 19 as obvious over Slivka and Angelo.

E. Decision

Upon consideration of the appeal, and for the reasons given herein, it is

ORDERED that the decision of the Examiner rejecting claims 1-4, 7-10 and 13-22 under 35 U.S.C. § 103(a) as unpatentable over Slivka and Angelo is affirmed.

ORDERED that the decision of the Examiner rejecting claims 11-12 under 35 U.S.C. § 103(a) as unpatentable over Slivka and Angelo is reversed.

No time period for taking any subsequent action in connection with this appeal may be extended under 37 C.F.R. § 1.136(a).

AFFIRMED-IN-PART

MAT

Michael T. Sanderson, ESQ.
King & Schlicki, PLLC
247 North Broadway
Lexington, KY 40507